UNIVERSITÉ DE MONTRÉAL

CENTRE DE RECHERCHE SUR LES TRANSPORTS

CASE POSTALE 6128, SUCCURSALE A, MONTRÉAL, QUÉBEC, H3C 3J7

SHORTEST PATH WITH TIME WINDOWS PROBLEM

by

Jacques Desrosiers*, Michel Sauvé**
and François Soumis***

* Service de l'enseignement des méthodes quantitatives,
Ecole des Hautes Etudes Commerciales

** Centre de recherche sur les transports, Université de Montréal

*** Département de mathématiques appliquées, École Polytechnique and
Centre de recherche sur les transports, Université de Montréal

Université de Montréal
Centre de recherche sur les transports - Publication #389
Juin 1985

Dépôt légal - 2e trimestre 1985

1. INTRODUCTION

Consider a network $G = (N:A)$ where $N = \{1,\ldots,n\}$ is the set of nodes and A is the set of oriented arcs (i,j). With each node i is associated a time interval $[a_i,b_i]$ during which this node must be visited. A cost $c_{ij} \in \mathbb{R}$ and a duration $t_{ij} > 0$ are associated with each arc $(i,j) \in A$. The set A is made up of all arcs (i,j) satisfying the scheduling constraints i.e. arcs for which $a_i + t_{ij} \leq b_j$.

The formulation of the least cost path problem with time windows on the nodes contains binary flow variables $x_{ij}, (i,j) \in A$, and continuous time variables $t_i \in [a_i,b_i]$, $i \in N$. The least cost path from s to t satisfying the scheduling constraints is given by the solution to the following problem :

$$\text{Min} \sum_{(i,j)\in A} c_{ij}\, x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = \begin{cases} 1 & i=s, \\ 0 & i\neq s, \neq t, \\ -1 & i=t \end{cases} \tag{2}$$

$$x_{ij} \geq 0 \qquad (i,j) \in A \tag{3}$$

$$a_i \leq t_i \leq b_i \qquad i \in N, \tag{4}$$

$$x_{ij} > 0 \Rightarrow t_i + t_{ij} \leq t_j \quad (i,j) \in A. \tag{5}$$

The problem formed by equations (1)-(3) is the classical shortest path problem between nodes s and t. Equations (4) and (5) impose the scheduling constraints. The conditional relation (5) imposes compatiblity between the flow variables x_{ij} on the arcs and the time variables t_i on the nodes. If the least cost path from s to t passes through the node i, then t_i is the time at which node i is visited; note that waiting is allowed at node i if arrival there occurs before a_i. If node i is not

part of the least cost path, then t_i is not constrained by (5) and may take any value in the interval $[a_i,b_i]$.

If there exists a feasible path in time from s to t, it is shown in [2] that the least cost path problem with time window constraints on the nodes always has an optimal integer solution.

This article describes the main results previously published in French in [2] and presents a FORTRAN implementation of the algorithm.

2. APPLICATION

This research on the least cost path problem with scheduling constraints on the nodes was originally inspired by the following application [3,4].

Consider a set of N tasks; a duration d_i and a time t_i in the interval $[a_i,b_i]$ are associated with each task i. Between the end of task i and the beginning of task j, we have a transition of duration t_{ij} and cost c_{ij} (including respectively the duration and cost of task j). The problem consists of covering the entire set of tasks while satisfying scheduling constraints and minimizing total cost. Without scheduling constraints, this problem can be formulated as a network problem, or as a set covering problem. In the latter case, we may proceed by column generation where the subproblem is a shortest path problem with positive or negative costs on the arcs. When scheduling constraints are included, the set problem becomes a shortest path problem with time window constraints on the nodes.

3. AN ALGORITHM FOR THE SHORTEST PATH PROBLEM WITH SCHEDULING CONSTRAINTS

3.1 Ford-Bellman-Moore Algorithm [5,1,6]

We briefly describe the Ford-Bellman-Moore algorithm before introducing our generalization of it. Denote by $F(i)=\{j \mid (i,j) \in A, j \in N\}$ the set of successors of i. The shortest path tree from the source s to all other nodes of the network is obtained by assigning a label to each node. The labels are then progressively improved. Let L be the set of nodes which have not been explored after an improvement : initially, L contains $\{s\}$ with the label $\pi_s = 0$. The labelling method consists of choosing a node $i \in L$, removing it from L and then attempting to improve all its successors D(i). The new improved nodes are inserted into L. When the set L has been exhausted, the label π_i of node i gives the least cost path from the source s.

This dynamic programming type approach has been adapted to handle the shortest path problem with scheduling constraints on the nodes. In the discussion below, we describe successively the nature of the labels, the optimality principle and label management techniques.

3.2 Nature of the labels

To simultaneously handle the time variables t_i and cost variables π_i associated with each node i, we use the pairs (t_i, π_i) as labels. To explain the meaning of these labels, we introduce the notion of a "feasible path" i.e. a path to which it is possible to associate times at nodes in such a way that the scheduling constraints are satisfied.

Note that for a given feasible path, there generally exists an infinite number of possible times satisfying the scheduling constraints. However we will treat this as a single path. Thus label (t_i, π_i) is assigned to node i if and only if there exists a feasible path of cost

π_i beginning at s and arriving at i at time t_i at the earliest. Such a path X_{si} associated with the label (t_i,π_i) has the same cost π_i for all arrival times within the interval $[t_i,b_i]$ as no cost is associated with waiting time.

The Ford-Bellman-Moore algorithm for the shortest path problem assigns a single label to each node which is then progressively improved. With the pairs (t_i,π_i), it is not possible to store only the best label for each node as there is no total ordering in $\mathbb{R}^2$: a set of labels $(t^\alpha_i,\pi^\alpha_i)$, $\alpha > 1$ must be stored. This set is obviously finite as the finite number of nodes means that there is only a finite number of feasible paths from s to i, and each path can only produce a single label.

3.3 Optimality principle

The success of labelling methods for the classical shortest path problem relies on the application of the following optimality principle : if X_{sj} is an optimal path between s and j of cost π_j and if arc (i,j) is the last arc in this path, then the subpath from s to i with cost $\pi_i = \pi_j - c_{ij}$ is optimal between s and i.

In the case of the shortest path with scheduling constraints on the nodes, the optimality principle has to be modified as follows. Let $(t^\alpha_j,\Pi^\alpha_j)$ be a label at node j and let X^α_{sj} be the associated path. If this path is optimal from among all paths arriving at node j at times t_j respecting $t_j < t^\alpha_j$, and if (i,j) is the last arc of X^α_{sj}, then subpath from s to i with costs $\pi_i = \pi^\alpha_j - c_{ij}$ is an optimal path from among all paths arriving at node i at times t_i respecting $t_i < t^\alpha_j - t_{ij}$.

3.4 Ordering relation on labels

We now construct a set of labels at each node i which give the best costs at different times in the interval $[a_i,b_i]$. We define the partial ordering relation $<$ at node i as follows :

$$(t_i^1,\pi_i^1) < (t_i^2,\pi_i^2) \iff (t_i^1 < t_i^2) \text{ and } (\pi_i^1 < \pi_i^2) \qquad (6)$$

A label is said to be "minimal" in this set if there exists no smaller label according to this ordering relation. The labels along the optimal route from s to i are the minimal labels in the whole set. This can be proved by induction. In fact the optimal label (t_i^*,π_i^*) is minimal, and, applying the optimality principle, the intermediate labels of a shortest path satisfying scheduling constraints are minimal. This result allows us to reduce the set of labels at each node by removing those which are not minimal and by storing an irreducible subset in the case where waiting times are permitted. The irreducible set associated with a node i is ordered in strictly increasing time order : using the partial ordering relation (6), we see that the costs are then in strictly decreasing order.

The irreducible set associated with node i contains $\alpha(i)$ labels :

$$(t_i^\alpha,\pi_i^\alpha), \quad \alpha = 1,2,\ldots,\alpha(i)$$

with $\quad a_i < t_i^1 < t_i^2 < \ldots < t_i^{\alpha(i)} < b_i \quad$ and

$$\pi_i^1 > \pi_i^2 > \ldots > \pi_i^{\alpha(i)}$$

For label α,π_i^α is the minimal cost from s to i within the time interval :

$$[t_i^\alpha,t_i^{\alpha+1}) \quad \text{if} \quad \alpha = 1,2,\ldots,\alpha(i)-1$$

or $\quad [t_i^{\alpha(i)},b_i]$ otherwise.

3.5 Description of the algorithm

Let $Q_i = (t_i^\alpha, \pi_i^\alpha)$, $\alpha = 1,2,\ldots,\alpha(i)$ be an irreducible set of labels associated with node i. By adding the arc (i,j) to the paths from s to i, we obtain a new set of labels at node j by applying :

$$F_{ij}(Q_i) = \bigcup_\alpha f_{ij}\{(t_j^\alpha, \pi_j^\alpha)\} \tag{7}$$

where

$$f_{ij}\{(t_i,\pi_i)\} = \begin{cases} \{(\max [a_j, t_i+t_{ij}],\ \pi_i+c_{ij})\} & \text{if } t_i+t_{ij} < b_j \\ \quad \text{or} & \\ \emptyset \quad \text{otherwise} & \end{cases} \tag{8}$$

The definition of f_{ij} allows waiting time at node j, and the new set $F_{ij}(Q_i)$ is not necessarily irreducible as the time values t_j^α may pile up at the value a_j for the first label :

$$t_j^\alpha = a_j \quad \text{for all } \alpha \text{ such that } t_i^\alpha + t_{ij} < a_j$$

Denote by $\overline{F_{ij}(Q_i) \cup Q_j}$ the reduction of the two sets retaining only the minimal labels.

The least cost path algorithm on a network with scheduling constraints on the nodes is similar to the Ford-Bellman-Moore algorithm.

0 : Initialization

$Q_s = \{(t_s^1 = a_s,\ \pi_s^1 = 0)\}$

$Q_i = \{(t_i^1 = a_i,\ \pi_i^1 = \infty)\}$ for all $i \neq s$

$L = \{s\}$

1 : Exploration of F(i)

Choose a node : $i \in L$

for all $j \in F(i)$ such that $\overline{F_{ij}(Q_i) \cup Q_j} \neq Q_j$

Set $Q_j = \overline{F_{ij}(Q_i) \cup Q_j}$

$L = L \cup \{j\}$

2 : Reduction of L

$L = L - \{i\}$

if $L = \emptyset$ then stop

otherwise return to 1.

3.6 Implementation strategies

L is a set of unexplored nodes after a modification. The next node to be explored is chosen according to the D'Esopo-Pape [2, 8] strategy. The set L is represented by a list accessible at its two extremities (i.e. a double-ended queue). The nodes to be explored are withdrawn from one end (called the "head"). The D'Esopo-Pape strategy consists of inserting the nodes reached for the first time at the "tail" of the list and those previously reached at the head of the list. Nodes already in the list are ignored. In this way, we hope to compute the optimal subpaths before lengthening them.

In an acyclic network corresponding to a set of partially ordered tasks (i.e. a network of tasks with a fixed schedule) the nodes can be numbered in such a way that if :

$$(i,j) \in A \quad \text{then} \quad i < j. \tag{9}$$

and their exploration in this order ensures that each node is inserted

into the list L once and only once. In our real-life examples where the width of the time windows does not exceed 30% of the horizon, the numbering of nodes in chronological order ensures that (9) is satisfied by the majority of arcs and the exploration of nodes in this order significantly reduces reinsertions and computation times.

The labels assigned to a node are placed in a list in increasing time order. $\overline{F_{ij}(Q_i) \cup Q_j}$ is calculated at node j by a sequential update of the set Q_j based on Δ_i, a subset of Q_i. Δ_i contains the labels which have been modified since i appeared in L : these are the only ones likely to generate modifications at node j. The update is carried out by working through the sets Δ_i and Q_j and simultaneously comparing the cost of usable paths from s to j.

3.7 Multiple visits to a single node

When the network contains negative cycles, the solution method described above does not guarantee paths without cycles. A path may pass through the same node j at several different times using different labels from the set Q_j. The number of cycles is however limited as they must all be carried out within the bounded interval $[a_i, b_i]$. When negative cycles exist, the complexity of the problem of constructing a shortest path without cycles is similar to that of the travelling salesman problem.

In the application discussed in Section 2, cycles are progressively eliminated using an implicit enumeration algorithm on the master problem. The scheduling constraints and the cost structure are such that cycles does not appear frequently in the solution of the set covering problem.

4. NUMERICAL RESULTS

The main factor explaining computer times is without doubt the number of arcs A of the network, to which can be added the multiplier effect of the division of labels and the number I of reinsertions into the modified node list. In general, the computation time T for the solution of a shortest path problem is proportional to the number of arcs |A| plus the number of new explorations of these arcs :

$$T = O(|A| + I \frac{|A|}{n}) \qquad (10)$$

In the case of the shortest path with scheduling constraints on the nodes, the effect of the division of nodes must be introduced. If E/n denotes the average number of labels per node during solution, the previous relation becomes :

$$T = O(\frac{E}{n} |A| + I \frac{E}{n} \frac{|A|}{n}) \qquad (11)$$

Computation times increase exponentially with the width of time intervals. When the width of the windows increases, the number of arcs in the network satisfying the relation

$$a_i + t_{ij} \leq b_j$$

increases, the subdivision of nodes is finer and the acyclic structure of the network is gradually destroyed. Although the computational effort increases with the width of time windows, the algorithm can handle large problems with time windows of width up to 30% of the time horizon. The networks handled in our application are constrained and almost acyclic, thus avoiding the proliferation of labels at each node and reducing the number of updates of the labels at a node.

5. CONCLUSION

The efficient treatment of the shortest path problem with time windows has enabled us to solve vehicle routing problems with scheduling constraints. Large problems of this type must be handled by decomposition so that maximum advantage may be taken of efficient network algorithms. Until now, these decomposition approaches separated the additional constraints from the network problem. With this approach, it is very difficult, based on the integer solutions of the network problem, to construct integer solutions satisfying the additional constraints as well. By solving the shortest path problem with scheduling constraints, we obtain an integer solution which satisfies some of the additional constraints as well as the network constraints. It is then easy to find an integer solution satisfying the entire set of constraints.

REFERENCES

[1] BELLMAN, R.E., On a routing problem. Quart. Appl. Math 16, (1958), 87-90.

[2] DESROSIERS, J., PELLETIER, P. and SOUMIS, F., Plus court chemin avec contraintes d'horaires, RAIRO 17, (1983), 357-377.

[3] DESROSIERS, J., DUMAS, Y. and SOUMIS, F., The multiple vehicles many to many routing problem with time windows , Publication #362, Centre de recherche sur les transports, Université de Montréal (1984).

[4] DESROSIERS, J., SOUMIS, F. and DESROCHERS, M., Routing with time windows by column generation, Networks 14, (1984), 545-565.

[5] FORD, L.R. Jr., Network flow theory, The Rand Corporation, Vol. 293 (1956).

[6] MOORE, E.F., The shortest path through a maze, Proceedings of the International Symposium on the Theory of Switching, Part II, April 2-5, 1957. Harvard University Press, Cambridge, Mass. (1959).

[7] PALLOTTINO, S., Adaptation de l'algorithme de D'Esopo-Pape pour la détermination de tous les chemins les plus courts, améliorations et simplifications. Publication #136, Centre de recherche sur les transports, Université de Montréal (1979).

[8] PAPE, U., Shortest path lengths. ACM Transaction on Mathematical Software 6 (1980), 451-455.